

String Vector based KNN for Automatic Text Summarization based on Text Categorization

Taeho Jo
President
Alpha AI Research
Cheongju, South Korea
tjo018@naver.com

Abstract—In this research, we propose the string vector based KNN version for automating the text summarization with its combination with the text classification. In the previous work, we proposed the string vector-based version as an approach to the text summarization, but must present the domain as a tag, manually. This research adopts the KNN algorithm which receives a string vector as the input data, as the approach to the text summarization and the text classification. In the proposed system, only a text without its domain is given as the input, the domain is automatically assigned to the text by the text classification, and each paragraph is classified into summary or non-summary by the classifier which corresponds to the domain. The goal of this research is to complete the automation of the text summarization by introducing the text classification and to improve the performance of both tasks, using the string vector based KNN algorithm.

I. INTRODUCTION

The text summarization is the process of extracting the essential part from a text as its summary. The essential part is regarded as some among the paragraphs in the text, and the text summarization is converted into a binary classification of paragraphs into summary or non-summary. The process of executing the text summarization is to partition a text which is given as the input into paragraphs, classify them into one of the two categories, and extract paragraphs which are classified into summary. The binary classification which is mapped from the text summarization belongs to the domain dependent classification where same paragraph may be classified differently depending on the domain. In advance, the domain should be presented as the additional input for selecting the classifier which corresponds to the domain.

It is required to label a text manually with its own domain for doing the text summarization in the previous work. The text summarization is defined into an independent binary classification domain by domain; a classifier is allocated to each domain. We need to know the domain of the input text for finding a corresponding classifier. The process of assigning a domain to a text is automated through the text classification, so only a text is provided for the text summarization system. We connect the text summarization which is covered in this research with the text classification.

The idea of this research is to connect the text summarization with the text classification and apply the string vector based KNN algorithm to both tasks. We define the similarity metric between two string vectors and modify the KNN algorithm into the version whose input is given as a string vector, using the similarity metric. The string vector-based version is applied to the text classification which decides the domain automatically to the input text. In the process of text summarization, a text is partitioned into paragraphs, each of them is classified into summary or non-summary using the string vector based KNN algorithm, and ones which are classified into summary are extracted as the final output. In this research, we propose the string vector based KNN algorithm as the approach to both the text summarization and the text classification.

Let us mention some benefits from this research. Encoding texts into string vectors is the solution to the problems in encoding them into numerical vectors. The chance of modifying other machine learning algorithms into string vector-based versions is opened by modifying the KNN algorithm so. The performance of both the text summarization and the text classification is improved by adopting the string vector based KNN algorithm as the approach to both tasks. The process of deciding the domain for the text summarization is automated by connecting it with the text classification.

Let us mention the remaining tasks for improving this research. We may define and characterize mathematically more semantic operations on strings. Paragraphs and text may be compound representations, each of which consists of more than one structured form. Other machine learning algorithms and deep learning algorithms are modified into string vector-based versions. The text summarization system which includes the text classification module may be implemented as a real program or a library by adopting the string vector based KNN algorithm.

II. PROPOSED WORKS

A. Encoding Proccerss

This section is concerned with the string vector as the text representation, and the similarity between string vectors. A string vector is a finite ordered set of strings; numerical

values are replaced by strings as the elements in a vector. It is required to define the similarity matrix for performing the operation on string vectors. The similarity between string vectors is the average over one-to-one similarities between their elements. This section is intended to describe the process of encoding a text into a string and the process of computing the similarity between string vectors.

The process of encoding a text into a string vector is illustrated in Figure 1. A text is indexed into a list of words and their information. The features of a string vector are defined as symbolic forms manually in the feature definition. A string which represents a text is filled with the words which corresponds to the features in the feature value assignment. Refer to [2] for studying the encoding process in more detail.

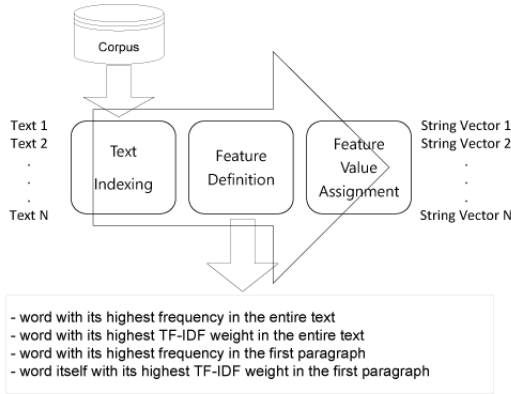


Figure 1. Process of Encoding Texts into String Vectors

The similarity matrix which is the basis for computing the similarity between two string vectors is illustrated in Figure 2. Each row and each column correspond to their own word, and each element in the similarity matrix is the similarity between two words, t_1 and t_2 , as shown in equation (1),

$$s_{ij} = \text{sim}(t_i, t_j) \quad (1)$$

The similarity between two words, t_i and t_j is computed by equation (2),

$$\text{sim}(d_i, d_j) = \frac{2\#(t_i \wedge t_j)}{\#(t_i) + \#(t_j)} \quad (2)$$

where $\#(t_i \wedge t_j)$ is the number of texts which include both t_i and t_j , $\#(t_i)$ is the number of texts which includes t_i , and $\#(t_j)$ is the number of texts which includes t_j . The value of similarity between two words, t_i and t_j , is always given as a normalized value between zero and one, as shown in equation (3),

$$0 \leq \text{sim}(t_i, t_j) \leq 1 \quad (3)$$

The similarity matrix is used for computing the similarity between string vectors which represent texts as the reference table.

$$\begin{matrix} & \text{Word 1} & \text{Word 2} & \dots & \text{Word N} \\ \begin{matrix} \text{Word 1} \\ \text{Word 2} \\ \dots \\ \text{Word N} \end{matrix} & \begin{bmatrix} S_{11} & S_{12} & \dots & S_{1N} \\ S_{21} & S_{22} & \dots & S_{2N} \\ \dots & \dots & \dots & \dots \\ S_{N1} & S_{N2} & \dots & S_{NN} \end{bmatrix} \end{matrix} \quad s_{ij} = \frac{2 \times \#(\text{word}_i, \text{word}_j)}{\#(\text{word}_i) + \#(\text{word}_j)}$$

Figure 2. Semantic Similarity Matrix

Let us mention the process of computing the similarity between string vectors as a semantic similarity between texts. The two string vectors which represent texts are notated by $\text{str}_1 = [t_{11} \ t_{12} \ \dots \ t_{1d}]$ and $\text{str}_2 = [t_{21} \ t_{22} \ \dots \ t_{2d}]$. The similarity between two string vectors is computed by equation (4),

$$\text{sim}(\text{str}_1, \text{str}_2) = \frac{1}{d} \sum_{i=1}^d \text{sim}(t_{1i}, t_{2i}). \quad (4)$$

The similarity between elements, $\text{sim}(t_{1i}, t_{2i})$, is taken from the similarity which was mentioned above. The value which is generated by equation (4) is always given as a normalized value between zero and one as shown in equation (5),

$$0 \leq \text{sim}(\text{str}_1, \text{str}_2) \leq 1. \quad (5)$$

Let us make some remarks on the encoding process and the computation of the similarity between texts. A text is encoded into a string vector by the word-text association, the feature definition, and the feature value assignment. The similarity matrix is defined as the basis for computing the similarity between two string vectors. The similarity between two string vectors is computed by equation (4). In the future research, we consider representing a text into multiple string vectors.

B. String Vector based KNN Algorithm

This section is concerned with the string vector based KNN algorithm. It was initially proposed as the approach to the text summarization by Jo in 2018 [1]. The similarity metric between string vectors which was described in the previous section is used for computing the similarity between a novice string vector and a training example. The labels of its nearest neighbors are voted with their identical weights for deciding the label of a novice input. This section is intended to describe the initial version of KNN algorithm from which its variants are derived.

The process of computing the similarity between a novice item and a training example is illustrated in Figure 00. The labeled words which are gathered as samples are encoded into string vectors, and they are notated by a set, $Tr = \{\text{str}_1, \text{str}_2, \dots, \text{str}_N\}$. A novice word is encoded into a string vector notated by str .

Its similarities with the string vectors which represent the sample words are computed by equation (??), as $sim(\mathbf{str}, \mathbf{str}_1), sim(\mathbf{str}, \mathbf{str}_2), \dots, sim(\mathbf{str}, \mathbf{str}_N)$. Some training examples which are most similar as the novice input are selected as its nearest neighbors.

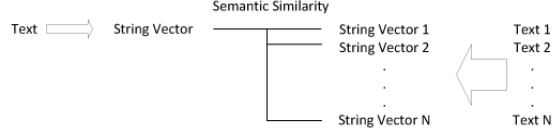


Figure 3. Similarities of Novice Input with Training Examples

In Figure 4, the process of selecting nearest neighbors by the ranking selection is illustrated. The training examples are sorted by the descending order of their similarities, after computing their similarities with a novice example. The training examples with their higher similarities with a novice example are selected as its nearest neighbors, as expressed in equation (6),

$$Nr = Select_k(Tr, \mathbf{str}), Nr \subseteq Tr \quad (6)$$

The set of nearest neighbors, Nr , is composed with elements as expressed in equation (7),

$$Nr = \{\mathbf{str}_1^{near}, \mathbf{str}_2^{near}, \dots, \mathbf{str}_k^{near}\} \quad (7)$$

The elements in the set, Nr , are used for voting their labels in the next step.

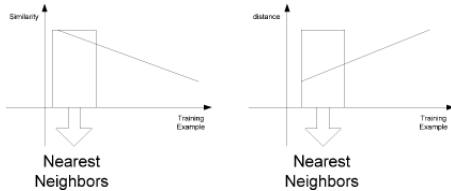


Figure 4. Selection of Most Similar or Least Distant Training Examples as Nearest Neighbors

The process of voting the labels of nearest neighbors for decoding the label of a novice word is illustrated in Figure 5. The predefined categories are notated by $c_1, c_2, \dots, c_m$, and the label of a nearest neighbor is notated by $c_j = label(\mathbf{str}_i^{near})$. The number of nearest neighbors which belong to the category, c_j , is notated by $Count(Nr, c_j)$. The label of the novice item, $\mathbf{str}$, is decided by equation (8),

$$label(\mathbf{str}) = \arg\max_{j=1}^m Count(Nr, c_j) \quad (8)$$

The process of deciding the label of a novice item by equation (8), is called voting.

Let us make some remarks on the string vector based KNN algorithm. It computes the similarities of a novice item with the training examples. The training examples with their highest similarities with the novice item are selected as its



Figure 5. Voting of Labels of Nearest Neighbors for Deciding Label of Novice Input

nearest neighbors. The label of the novice item is decided by voting the labels of its nearest neighbors. The ranking selection is adopted for selecting the nearest neighbors from training examples, in this version.

C. System Architecture

This section is concerned with the text summarization system which adopts the string vector based KNN algorithm. In Section II-B, we described the proposed version of KNN algorithm as the approach to the text summarization. It is viewed as a binary classification which classifies a paragraph into summary or non-summary. This section is intended to describe the text summarization system with respect to its function and its architecture.

The process of sampling paragraphs domain by domain is illustrated in Figure 6. Even if it is possible map the text summarization into an instance of text classification, a paragraph may be classified differently depending on the domain. Sample paragraphs which are labeled with summary or non-summary are gathered domain by domain. The text which is given as the input is tagged with a domain, and the classifier which corresponds to the domain is appointed. The sample paragraphs are encoded into string vectors in each domain.

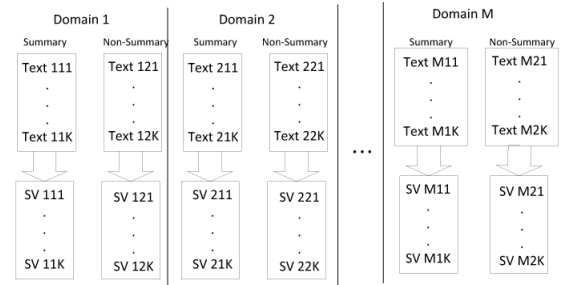


Figure 6. Preparation of Training Examples

The entire architecture of the proposed text summarization system is illustrated in Figure 7. A text is given as the input, and paragraphs are extracted by partitioning the text. In the encoding module, the sample paragraphs in the summary group and the non-summary group and ones which are extracted from the text are mapped into string vectors in the encoding module. The paragraphs from the text are classified into one of the two categories in the similarity computation module and the voting module. The paragraphs which are

classified into summary are extracted as the output in this system.

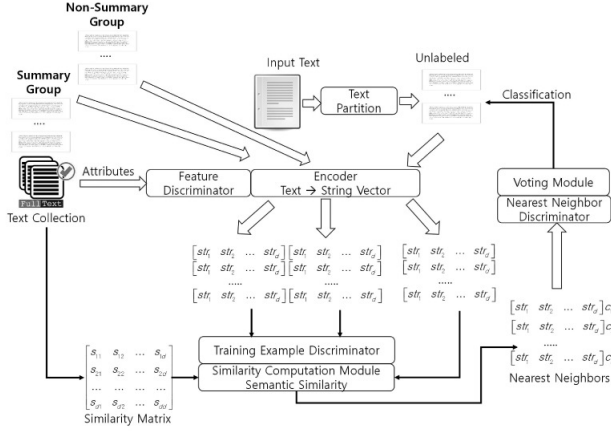


Figure 7. System Architecture

The execution flow of the text summarization system is illustrated in Figure 8. In each domain, sample paragraphs which are labeled with summary or non-summary are gathered. The input text is partitioned into novice paragraphs, and both sample paragraphs and novice paragraphs are encoded into string vectors. Each paragraph is classified into summary or non-summary, and there are two groups of paragraphs in the text: summary group and non-summary group. The paragraph which are classified into summary are extracted as the summary of the input text in this system.

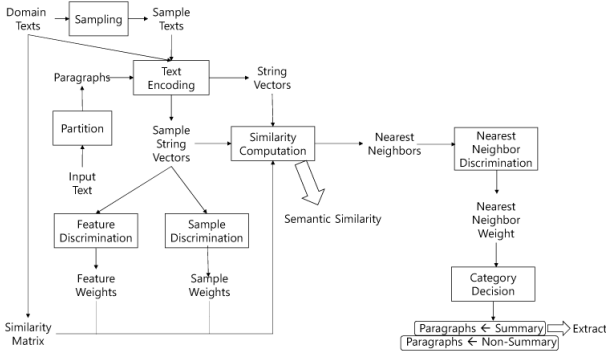


Figure 8. System Flow

Let us make some remarks on the proposed system which is illustrated in Figure 7 as its architecture. The text summarization is defined as the binary classification where each paragraph is classified into summary or non-summary. Paragraphs are encoded into string vectors instead of numerical vectors, and they are classified directly. Ones which are classified with summary are selected as the summary of the input text. We need to add the text classification module for predicting the domain of input text automatically.

D. Connection with Text Classification

This section is concerned with the connection of the text summarization with the text classification. In the previous section, we mentioned the text summarization system as an instance of domain dependent classification. The domain is automatically decided for nominating a classifier by connecting the keyword with the text classification. The KNN algorithm which was described in Section II-B is used for implementing the text classification. This section is intended to describe the connection of the text summarization system with the text classification for automating the decision of the domain.

The system architecture of the text categorization system which consists of the encoding module, the similarity computation module, and the voting module is illustrated in Figure 9. The encoding module is for encoding texts into string vectors by the process which is described in Section II-A. The similarity computation module is for computing the similarities between string vectors which represent a novice text and a sample text and selecting ones with their highest similarities as the nearest neighbors. The voting module is for deciding the label of the novice item by voting the labels of the nearest neighbors. This system used for automating deciding the domain of the input text.

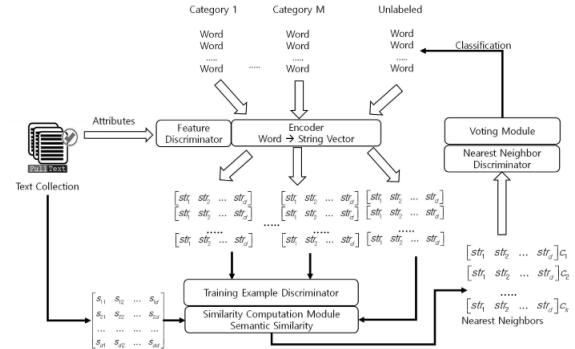


Figure 9. Text Categorization System

The connection of the text summarization with the text classification is illustrated in Figure 10. The texts each of which is labeled with its own domain are prepared as the sample texts, and the approach to the text classification is trained with them. A domain is assigned to a novice text by classifying it into a domain automatically. The novice text is provided with its domain to the text summarization system which is described in Section II-C. The role of the text classification system is to nominate the classifier which corresponds to the domain for the text summarization, automatically.

The process of executing the text summarization, connecting it with the text classification is illustrated in Figure 11. The sample texts each of which is labeled with its

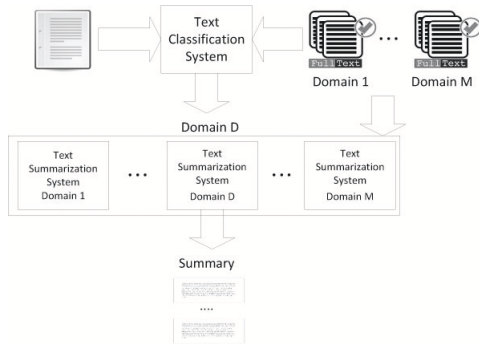


Figure 10. Text Summarization System connected with Text categorization

own domain are prepared, and the sample paragraphs each of which is labeled with summary or non-summary are prepared in each domain. A novice text is classified into one among the predefined domains, and the classifier which corresponds to the domain is nominated. The novice text is partitioned into a list of paragraphs, and each paragraph is classified into summary or non-summary. The paragraphs which are labeled with summary is the final output from this system; the domain which is assigned to the input text is the guide for selecting a classifier.

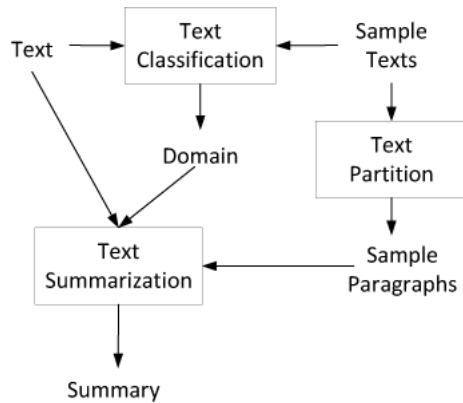


Figure 11. System Flow of Text Summarization connected with Text Categorization

Let us make some remarks on the connection of the text summarization with the text classification. The text classification is applied to the automatic decision of the input text domain. The role of text classification is to decide the domain for nominating a classifier, and the role of the text summarization is to extract important paragraphs as the summary. In the execution flow, the input text is classified into a domain, it is partitioned into a list of paragraphs, and each paragraph is classified into summary or non-summary. We consider connecting the text classification as the main task the text summarization as the opposite to what is proposed in this research.

REFERENCES

- [1] T. Jo, "Automatic Text Summarization using String Vector based K Nearest Neighbor", 6005-6016, Journal of Intelligent and Fuzzy Systems, 35, 2018.
- [2] T. Jo, "Semantic String Operation for Specializing AHC Algorithm for Text Clustering", 1083-1100, Annals of Mathematics and Artificial Intelligence, 2019.